Internet of Things 22 (2023) 100773

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Smart home anomaly-based IDS: Architecture proposal and case study

Agustín Lara [a], Vicente Mayor [a,*], Rafael Estepa [a], Antonio Estepa [a], Jesús E. Díaz-Verdejo [b]

[a] *Department of Telematics Engineering, University of Seville, C/ Camino de los Descubrimientos s/n, Seville, 41092, Spain*
[b] *Department of Signal Theory, Telematics and Communications, University of Granada, C/ Periodista Daniel Saucedo Aranda s/n, Granada, 18071, Spain*

ARTICLE INFO

Keywords:
IDS
A-IDS
Anomaly detection
Smart home
IDS architecture proposal

ABSTRACT

The complexity and diversity of the technologies involved in the Internet of Things (IoT) challenge the generalization of security solutions based on anomaly detection, which should fit the particularities of each context and deployment and allow for performance comparison.

In this work, we provide a flexible architecture based on building blocks suited for detecting anomalies in the network traffic and the application-layer data exchanged by IoT devices in the context of Smart Home. Following this architecture, we have defined a particular Intrusion Detector System (IDS) for a case study that uses a public dataset with the electrical consumption of 21 home devices over one year. In particular, we have defined ten Indicators of Compromise (IoC) to detect network attacks and two anomaly detectors to detect false command or data injection attacks. We have also included a signature-based IDS (Snort) to extend the detection range to known attacks. We have reproduced eight network attacks (e.g., DoS, scanning) and four False Command or Data Injection attacks to test our IDS performance. The results show that all attacks were successfully detected by our IoCs and anomaly detectors with a false positive rate lower than 0.3%. Signature detection was able to detect only 4 out of 12 attacks. Our architecture and the IDS developed can be a reference for developing future IDS suited to different contexts or use cases. Given that we use a public dataset, our contribution can also serve as a baseline for comparison with new techniques that improve detection performance.

1. Introduction

In the past decade, the Internet of Things (IoT) has experienced significant growth in several domains, such as Industry 4.0 [1], Smart Cities [2], or Smart Homes [3]. But this success has also raised new security challenges, including designing software architectures and security models that address cyber threats in all IoT layers (e.g., application, network, middleware, sensing, etc.) [4]. The heterogeneity of devices, vulnerabilities, and technologies involved in the IoT challenges the adoption of generic cyber security solutions [5,6], advising the customization of defensive systems to specific domains and applications.

The context of this work is Smart Home. Undoubtedly, there is a growing presence and usage of smart devices within home. Connected homes have become pervasive today as people demand the comfort and functionalities of smart devices such as smart speakers, smart TVs, surveillance webcams, or smart plugs, to name a few. But the practicality of such devices does not come without

* Corresponding author.
E-mail addresses: alarar@us.es (A. Lara), vmayor@us.es (V. Mayor), rafaestepa@us.es (R. Estepa), aestepa@us.es (A. Estepa), jedv@ugr.es (J.E. Díaz-Verdejo).

https://doi.org/10.1016/j.iot.2023.100773
Received 12 January 2023; Received in revised form 23 March 2023; Accepted 27 March 2023
Available online 30 March 2023
2542-6605/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

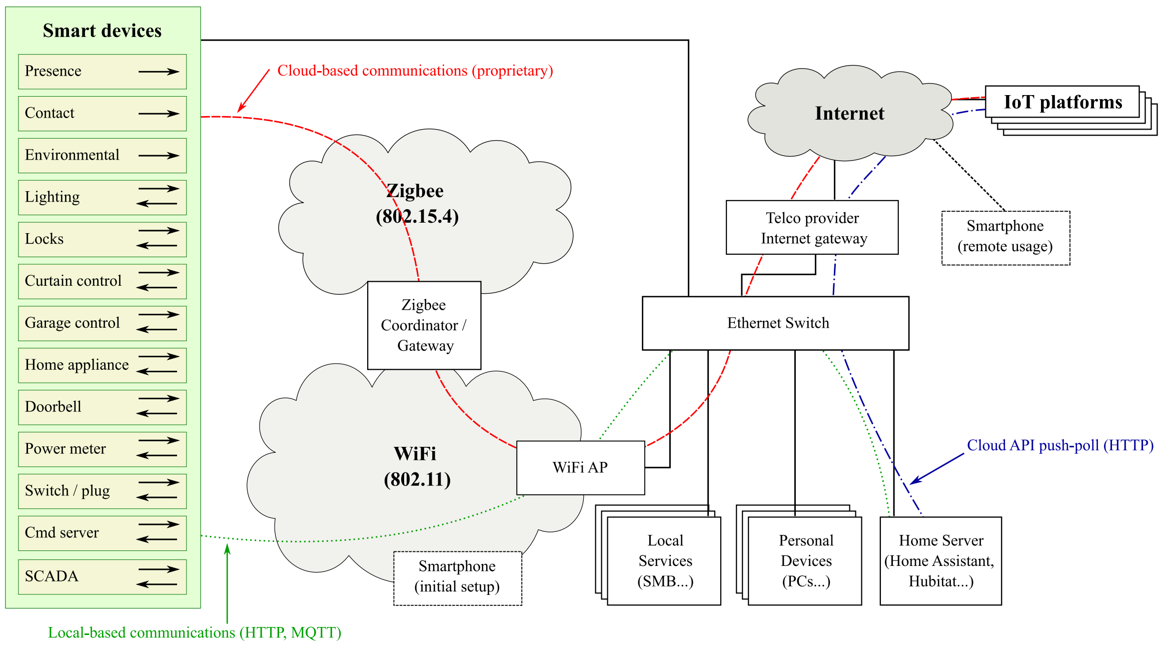
Smart devices
Presence
Contact
Environmental
Lighting
Locks
Curtain control
Garage control
Home appliance
Doorbell
Power meter
Switch / plug
Cmd server
SCADA
Cloud-based communications (proprietary)
Zigbee (802.15.4)
Zigbee Coordinator / Gateway
WiFi (802.11)
WiFi AP
Smartphone (initial setup)
Internet
IoT platforms
Telco provider Internet gateway
Smartphone (remote usage)
Ethernet Switch
Cloud API push-poll (HTTP)
Local Services (SMB...)
Personal Devices (PCs...)
Home Server (Home Assistant, Hubitat...)
Local-based communications (HTTP, MQTT)

Fig. 1. Main elements and networks in a smart home generic environment.

risk as, with billions of users, smart devices have become one of the most lucrative targets for cyber criminals [7]. According to [8], there are an estimated 175 million smart homes worldwide, and 40.8% of them have at least one device vulnerable to cyber attacks that put the whole home in danger. Indeed, an estimated 35% of US broadband households dealt with a data security incident in 2021. A compromised smart device not only threatens domestic life in many ways [9] but also enables hackers to use the home network resources to turn it into a botnet to launch a cyber-attack on national infrastructures. A notable example is the rooftop solar power system installed in millions of residential buildings worldwide and its interconnection with the smart grid generation domain [10,11]. Thus, due to its impact on domestic life and global security, there is a clear need for detecting cyber attacks in the context of a smart home [12,13].

Intrusion Detection Systems (IDS) are central for detecting security incidents in any organization [14], including the home. IDS are commonly deployed in the target environment, monitoring the events occurring in the hosts or network, analyzing them for signs of possible incidents. Unfortunately, there is no widely accepted reference architecture for a smart home (albeit a comprehensive proposal can be found in [15]), and authors tend to use high-level examples such as in [16] or [17]. However, for tractability, we provide in Fig. 1 a non-exhaustive smart home reference scenario based on the authors' observation.[1]

A smart home may include several IoT devices that can be classified as (a) sensors, for periodic monitoring of a variable (e.g., temperature, fan speed), (b) actuators, that receive commands to accomplish an action (e.g., turn on a light, close a door), or (c) both (e.g., smart plug). Typically, users have devices from different manufacturers, each operating in a particular manner. Most households use WiFi (IEEE 802.11) or Zigbee (IEEE 802.15.4) as wireless communication technology [18]. Zigbee devices typically require a Zigbee Coordinator/Gateway for Internet connectivity (e.g., Philips Hue Bridge). Before operation, smart devices need an initial setup to configure, at least, their wireless connection and cloud-platform credentials, which is commonly done through a smartphone app offered by the manufacturer (e.g., Mi Home). Once configured, devices communicate with their provider's IoT platform, generally, through a proprietary protocol. Most devices provide a web/mobile application that allow users to track the device's state and provide access to the device's functionality. Some manufacturers also provide a token or credential-based API (e.g., API-Tuya) so that users can write code to manage their smart devices. With different apps managing different devices, one can recur to a Home Server (e.g., Home Assistant) to centralize the management of multiple devices in a local database, enabling cross-platform automation flows via providers' REST APIs. The Home Server can also directly communicate with devices (e.g., Shelly, Sonoff) using standard protocols such as HTTP or MQTT. In this work, we assume the existence of a home server.

In a home environment such as the one described above, having an IDS in place to monitor signs of attacks is both crucial and challenging. And, although IDS have been around for decades in organizational contexts [19] (including IoT [20]), the development of IDS suited for smart homes still remains as an open research challenge [9,18]. Several reasons can be argued for this, but an evident impairment is the wide diversity of smart devices and technologies found in households, which challenges IDS design and deployment and translates to a wide attack surface [9,15,16]. This work aims to contribute to the challenge of developing IDS suited

[1] The reader is referred to [12,13] for a more complete list of technologies.

for smart homes by suggesting a reference IDS architecture that can be adapted to house-specific devices, technologies and threats. We also provide a case study for guidance. In our case study, we develop a customized IDS to detect common attacks in a scenario based on a public dataset. More specifically, the main contributions of this paper are:

- The analysis of current IDS proposals for smart home in light of the challenges identified in the literature.
- The definition of a flexible and modular architecture that can serve as a reference to design and implement IDS customized for a particular household's smart devices and threats.
- The realization of a particular IDS based on our architecture suited for detecting attacks in a smart home scenario produced from a public dataset.
- A comprehensive performance analysis of every component of the IDS designed using an in-house attack dataset (publicly available).

Our contributions can be used in two ways: (a) as a reference for designing defensive systems suited to different scenarios or attacks, (b) as a baseline to test new anomaly detection techniques or indicators that improve the performance of the defensive system (since we use a public dataset).

The remainder of the paper is as follows. Section 2 provides a review of attacks applicable to smart homes and the IDS suggested in the scientific literature to detect these attacks, emphasizing the differences between our proposal and existing solutions. Section 3 describes our generic reference architecture, which is used in the design of a IDS for a case use in Section 4. In Section 5 we provide a comprehensive performance evaluation of the IDS implemented. Section 6 shows the main limitations of this work. Finally, Section 7 concludes the paper.

2. State of the art in smart home IDSs

Smart home threats, challenges and security schemes have been addressed in the scientific literature from different perspectives. For example, the Smart Grid research field tends to focus on vulnerabilities and threats that impact the electricity generation domain and the home environment is commonly restricted to the electrical consumption of devices (and not their functionality) [13]. On the other hand, the IoT research field also addresses IDS in smart environments with a broader scope [21,22], only sometimes relevant to a smart home. In this paper, we restrict our analysis to a purely domestic environment.

2.1. Common attacks in the smart home

Cyberattacks on smart homes may impact several cyber–physical systems such as smart plugs, lighting controllers, heating, etc. (which today can also be remotely controlled by smart speakers), although new attack surfaces are emerging regularly [15,23]. A prevalent set of attack techniques used by adversaries is:

- Scanning. Scanning attacks gather information (by probing victim infrastructure via network traffic) that can be used during targeting. Examples of this technique in the context of a smart home can be found in [7,24–26].
- Sniffing. A technique in which adversaries may sniff network traffic to capture information about an environment, including authentication material passed over the network. Examples of sniffing attacks in a smart home can be found in [7,17,25–31].
- Data infection. Adversaries may insert, delete, or manipulate data in a smart device or server to influence external outcomes or hide activity. The type of modification and the impact it will have depends on the target application and process as well as the goals and objectives of the adversary. Data infection attacks in the context of a smart home can be found in [9,17,25,26,28–32]
- Command and control (C&C). This tactic includes techniques that adversaries may use to communicate with systems under their control within a victim network. Adversaries commonly attempt to mimic normal, expected traffic to avoid detection. Examples of (C&C) in the context of a smart home are [7,9,17,24,25,28,30,31].
- DoS. Denial of Service attacks aim to degrade or block users' access to targeted resources. Network DoS can be performed by exhausting the network bandwidth services rely on. Smart Home examples can be found in [7,17,24–26,28–31,33,34].
- Spoofing. Adversaries may force a device to communicate through an adversary-controlled system so they can collect information or perform additional actions. Examples of this technique in a smart home can be found at [7,9,17,25–31].
- Web attacks. An attacker may take advantage of a weakness in an Internet-facing web-server running at any smart device or home server using software, data, or commands in order to cause unintended or unanticipated behavior. Examples of this attack in a smart home can be found in [17,28,31].

The interested reader can find in [9] an excellent and up-to-date taxonomy of cyber threats in smart homes, attack vectors, and impact on the systems, domestic life and emotional consequences.

2.2. IDS in the context of smart home: challenges and related works

As stated earlier, detecting cyber incidents in connected homes is a challenging endeavor [35] and one requirement frequently found in the literature is to monitor the home network for signs of attacks [7,17]. IDS are key for this task.

IDS have been extensively studied in the literature [36]. IDS systems can be broadly categorized into two groups: Signature-based Intrusion Detection System (S-IDS) and Anomaly-based Intrusion Detection System (A-IDS) [19]. Whilst S-IDS are based on

Table 1
IDSs suggested for smart home.

Work	Method[1]	Location (arch)	Data input	Validation
[41]	A-IDS	Local (centralized)	Network	Empirical
[42]	A-IDS	Local (centralized)	Network (flows)	Empirical
[38]	S-IDS	Local (centralized)	Network	Simulation
[43]	A-IDS	Local + cloud	Network (flows)	Simulation
[40]	A/S-IDS	Local (centralized)	Logs	Empirical
[44]	A-IDS	Local (centralized)	Network (flows)	Empirical
[45]	A-IDS	Local (centralized)	Network (flows)	Empirical
[46]	A-IDS	Local (centralized)	Network (flows)	Simulation
[47]	A-IDS	Local (centralized)	Network (flows)	Empirical
[39]	A/S-IDS	Local + cloud	Network (cloud)	Empirical
[48]	A-IDS	Local (centralized)	Network (flows)	Empirical
[49]	A-IDS	Local (centralized)	Network (flows)	Simulation

pattern-matching techniques and can only detect known attacks, A-IDS are based on deviation from a model of normality and can notice 0-day attacks, which makes both techniques complementary. IDS can also be classified according to whether they monitor the network or host events (e.g., logs).

A plethora of IDS systems have been proposed in several application domains (including IoT [22]), but, to the best of our knowledge, only those in Table 1 have been proposed explicitly for smart home. Most of these IDS use anomaly-based detection techniques (A-IDS) over statistics on network traffic (flows) [37], although signature-based detection systems have also been proposed to a lesser extent either as standalone system [38,39], or in combination with A-IDS [39,40].

The comparison of the proposals in Table 1 is challenging due to their differences in techniques, scenarios, attacks, and validation processes. However, they can be analyzed in light of the following **challenges** identified in [50,51] for IDS development in the context of IoT:

- Feasible data collection. IDS placement, architecture and local home network technologies should be considered. Changes to the standard behavior of network devices should not be required.
- Attack detection range and technique variants. IDS should detect a wide range of attack types. Variants of each attack technique should ideally be considered.
- Reproducibility. The work should be reproducible by other researchers to assess IDS performance. Validation should ideally include real-life public datasets.
- Privacy. User data privacy should be preserved.
- Alert management. A mechanism should regulate IDS alerts sent to the user (i.e., ideally, few alerts and significant). Alerts should provide enough information to investigate the incident.
- Generalizable. The detection schemes should be generic and usable in scenarios different from the study.
- Computationally feasible. Anomaly-based techniques should be executable in real-time and avoid scalability issues.

Table 2 reviews the IDSs listed in Table 1 in light of the previous challenges. The conclusions could be summarized as follows:

- Data collection is frequently partial in the home network (e.g., only WiFi traffic) or unspecified [38,49], or it requires the modification of the standard behavior of network electronics [41,42,45,46].
- The attack types detected are different on each work, ranging between 2 and 4, being DoS the most prevalent type. Some works address network attacks (e.g., DoS, scanning) [41,45] while other address application-level attacks (e.g., False Command Injection) dependent of specific devices [48,49]. The number of techniques variations tested is generally low (2 in the best case).
- Some works pose constraints that prevent the application of their IDS in different scenarios (e.g., due to the use of vendor-specific APIs [40,48] or protocols [49]).
- Few works (only [39,41,46,49]) provide enough information to be replicated.
- Privacy issues are not generally addressed. Although if IDS data input and output are stored or handled locally, the need for privacy could be relaxed (if there is no third-party involvement).
- Few works (only [39,40]) include some scheme for alert management or discuss computational performance.

So it can be concluded that the intrusion detectors proposed for smart home in Table 1 fall short in fulfilling some of the previous challenges.

2.3. Originality

We believe that covering all the previous challenges would exceed the typical design of a single IDS. For this reason, our approach, rather than simply proposing a new IDS, is to provide a modular and flexible architecture that can be taken as a reference for the development of IDS suited to the challenges and specific scenario found in each house in terms of suitable detectors and deployment architecture. Most related works, however, focus exclusively on the techniques for attack detection.

Table 2
Related works comparison (IDS challenges).

Challenge	This paper	Related works											
		[41]	[42]	[38]	[43]	[40]	[44]	[45]	[46]	[47]	[39]	[48]	[49]
Feasible data collection	Centralized architecture (flows have to be aggregated into a device)	WiFi traffic is collected at the (modified) AP	Traffic is collected at the providers' gateway (to be modified)	Unspecified	Traffic is collected at the home/IoT gateway (to be modified), cloud components are needed	Devices data is gathered from third-party providers API, cloud components are required	It requires collective communications between smart homes. RPis are used to route traffic (bandwidth impact)	WiFi traffic is collected at the (modified) AP	The home router has to support SDN, and a local PC must implement a SDN controller	Centralized architecture (flows have to be aggregated into a device)	Traffic collection techniques are not specified, cloud components (SIEM) are needed	The home router has to support SDN, and a local PC must implement a SDN controller	Unspecified
Attack detection capability	6 categories (see Table 4)	3 categories (DoS, Scanning, Spoofing)	2 categories (Dos, E-DoS)	Attacks already detected by Snort/Suricata/Bro rules	Unspecified	3 categories (False Command Injection, DoS, Spoofing)	Unspecified	3 categories (Scanning, DoS, Spoofing)	4 categories (Scanning, DoS, Spoofing, Web Attacks)	No	Unspecified	2 categories (False command injection, web attacks)	3 categories (False command injection, DoS, energy)
Detecting different attack techniques	Up to 4 variants	At least 2 variants	At least 2 variants	Unspecified	Unspecified	No (APP-layer dependent)	Unspecified	Up to 2 variants	Yes	2 variants	Unspecified	Attacks are dedicated to a single device: Philips Hue	Unspecified
Validation and reproducibility	Yes	Yes	Missing resources and information	Non-representative dataset, missing information	Missing resources and information	Missing resources and information	Missing resources and information	Missing resources and information	Yes	Missing resources and information	Yes	Missing resources and information	Yes
Privacy	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified
Alerts management	Yes	No	No	No	No	Yes	No	No	No	No	Yes	No	No
Generalizable models	Partial (see limitations)	Supervised techniques require data labeling.	Yes	Yes	Yes	Dependent on manufacturers API, rules are scenario-dependent	Yes	Yes	Packet headers features are scenario-dependent	Yes	Rules are scenario-dependent	Limited to Phillips Hue devices	Limited to MQTT, BACnet and Modbus protocols
Feasible computational resources	Yes	Unspecified	Unspecified	Yes	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	No	Yes	Unspecified	Unspecified
Limitations	• Centralized architecture (flows aggregation, single point of failure) • Anomaly-based data module might require periodical training	• Access point needs modifications • Only WiFi smart devices are tested • Lacks computational performance analysis • It requires periodical training and data labeling (supervised ML)	• Provider's gateway needs modifications • Limited to network attacks (N-IDS) • Lacks computational performance analysis	• Proposed scenario is not particularized to a smart-home scenario • It provides a comparison between popular S-IDS software (not a new architecture) • Datasets are not particular to a smart-home scenario • Limited to network attacks	• Unclear characterization of anomalous traffic and attack coverage • Access point or IoT gateways require modifications • It relies on cloud infrastructure (privacy concern) • Restricted to network attacks • Lacks computational performance analysis	• It relies on providers' IoT platform APIs • It relies on cloud infrastructure (privacy concern) • Context-driven, it is unable to detect network attacks • Lacks of computational performance analysis	• Smart-homes collaborate each other for threat detection (privacy concern) • A device is required to route all the analyzed traffic, so bandwidth might be reduced • Limited to network attacks • Lacks computational performance analysis	• Access point needs modifications • Only WiFi smart devices are tested • Limited to network attacks • Lacks computational performance analysis	• Some network devices must support SDN • One of the chosen datasets (NSL-KDD) might not be representative for smart-home scenarios • Chosen features are scenario-dependent • Lacks of computational performance analysis	• Centralized architecture (flows aggregation, single failure point) • Only valid for DoS attacks • The testbed IDS was not able to operate in real-time	• It relies on cloud infrastructure (privacy concern) • Validation lacks some key metrics (performance, tested attacks)	• Some network devices have to support SDN or OpenFlow • Lacks of computational performance analysis • Testbed limited to Philips Hue devices	• Restricted to three protocols (it relies on app-layer features) • Lacks computational performance analysis

We also create a case study where we design and develop an IDS that follows our architecture and is customized to a particular case based on a public dataset with the electrical consumption of 21 devices in a smart home. The main differences between our IDS and those in Table 1 are shown in Table 2. In summary, our design focuses on anomaly detection techniques and is complemented by signature detection. We detect up to 6 types of attacks and up to 4 technique variants. Our system has the potential for alert management, including event correlation and classification of attacks. Finally, the input datasets used are public. As such, our work can be replicated or even improved by new variants suggested by the scientific community.

3. Proposed architecture

In this section, we provide a high-level reference architecture for a smart home IDS. This architecture should be generic, modular and flexible to allow one to adapt it to the specific requirements and resources available on each particular scenario. The modules described in this section will be refined and elaborated on in the case study presented in Section 4.

The system architecture is illustrated in Fig. 2 and encompasses four main modules:

- **Network-level module**. This module behaves as a Network Behaviour Analysis (NBA) system [14]. As such, it examines statistics on network traffic to identify unusual traffic flows, such as denial of service (DoS) attacks, certain forms of malware (e.g., backdoors), or policy violations (e.g., a home system providing network services to external systems). These events apply to most smart devices despite their functionality. A pre-requisite of this module is that network traffic from smart devices is collected and aggregated in the home network as it constitutes its data input.[2] The module takes a packet capture -pcap- file with the collection of IP packets sent or received by the smart devices during a period and carries out the following operations:

[2] In a reference environment like shown in Fig. 1, these packets can be captured using the port mirror feature of Ethernet switches assuming that a Home server is in place.

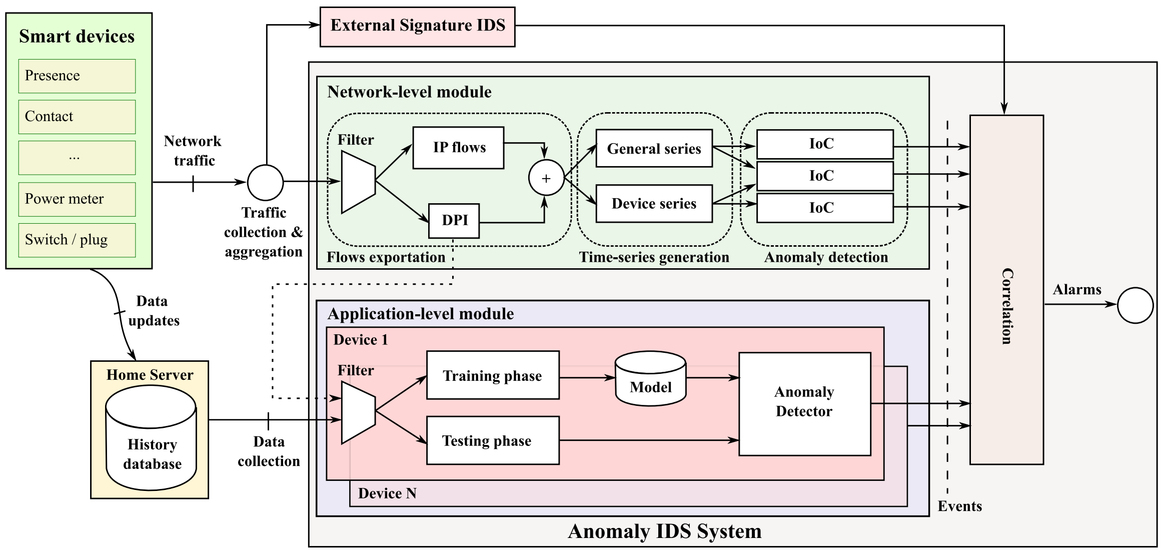

Fig. 2. Proposed IDS architecture.

1. Flows export. Generate the IP flows report related to the IPs under scrutiny (other IPs can be filtered). A common standard for this task is IP flow information export (IPFIX) [52]. If available, Deep Packet Inspection (DPI) can enhance this characterization.[3]
2. Time-series generation. The flows exported are processed to obtain statistics on network traffic. These statistics could be generic (e.g., overall number of ARP messages in the subnet) or device-specific (e.g., number of external IPs that requested a connection with a particular smart device). Since traffic flows are refreshed periodically, these statistics conform to time series related to generic network aspects or to specific devices.
3. Anomaly Detection. Each time series featured becomes the input to an indicator of compromise (IoC) designed to detect unusual behavior of a certain property. Each IoC generates an event after the detection of an anomaly. Several IoCs are defined in the literature revised in Section 2, and statistical methods are good candidates for online operation due to their simplicity [21]. However, the design of each IoC (i.e., feature and detection technique) will depend on the specific local scenario and the attacks one wants to detect. The next section provides a set of IoCs designed for our case study.

- **Application-level module**. This module behaves as a host-based IDS, monitoring the data exchanged between each smart device and the Home server during operation, seeking the discovery of anomalies produced by attacks such as False Command Injection or Data Manipulation.
  The input data source of this module is the historic dataset stored in the Home Server which can be accessed via API or directly from the database server. Detecting anomalies at this level requires learning a behavioral model that should be trained with real-life data, which implies that this should be a highly specialized module (i.e., on a per-device basics). A wide range of detection techniques [21,53,54] could be used for the design of this module that, in general, will examine the historic data from a smart device for anomalies with its past (device anomaly) or with respect to similar devices (group anomaly). Given the variability of the behavior over time, and the fact that anomalies may not be attacks but simply data errors or device malfunctioning [54], this module might show a high rate of false alarms, which should be considered in the design of the detector's point of operation and also in the correlation module.
- **Signature-based IDS module**. Conversely to the two previous modules, based on anomaly detection, this module is based on signature detection. Signature-based IDS (Snort, Suricata, Bro, ..) can be used in the context of smart home [55] to detect known attacks with a lower rate of false positive, complementing the job of anomaly detection at a lower computational cost.
- **Correlation module**. This module combines the events generated by the three previous modules and generate the alarms to be reviewed by a Security Operation Center (SOC) or the IDS operator (e.g., homeowner). Correlation allows one to gain knowledge from the events detected and reduce the number of alerts generated to a few significant ones (i.e., those likely to be attacks [56]). This is particularly important in a home environment where users may lack technical expertise. The alarms should include enough information to investigate the event and determine if further action is necessary. Some appropriate actions could be: ignoring the alarm, classifying the alarm as a false positive and exerting corrective actions

[3] Deep Packet Inspection (DPI) can: (a) enrich flows with extra information of interest (e.g., operating system fingerprint, geolocation, etc.); (b) parse the application-level dialog exchanged between IoT devices and the server (if the packets were not encrypted).

Table 3
Dataset devices.

Device ID	Description
B1	North Bedroom
B2	Master/South Br
BM	Basement Plugs&Lights
CD	Clothes Dryer
CW	Clothes Washer
DN	Dining Room Plugs
DW	Dishwater
EB	Electronics Workbench
EQ	Security/Network
FG	Kitchen Fridge
FR	HVAC/Furnace
GR	Garage
HP	HeatPump
HT	Instant Hot Water Unit
OF	Home Office
OU	Outside Plug
TV	Ent TV/PVR/AMP
UT	Utility Room Plug
WO	Wall Oven
WH	Whole-House Meter
UN	Unmetered Loads

to avoid future notifications in similar circumstances (e.g., re-training or modification of the configuration), or accepting the alarm but changing the period of transmission of similar alarms. This module is also responsible for filtering unwanted alarms.

On design choices The generic reference architecture in Fig. 2 can be used for designing IDSs adapted to specific contexts. Local decisions include the following aspects:

- Architecture. The number of modules in place and the IDS software architecture (e.g., distributed or centralized). For example, a distributed architecture could run each module in separate hardware, improving scalability.
- Network-level module. Design of the set of IoCs suited to the attacks to be detected.
- Application-level module. Design of anomaly detection techniques suited to each smart device profile, attacks to detect, and available data.
- Signature-based IDS. Selection of software and ruleset in place.
- Correlation. Design of correlation directives and alarm messages according to the potential events generated by the modules in place.

Finally, note that our reference architecture is also limited in some aspects (see Section 6) that could also impact design decisions.

4 Case study: IoT smart home

In this section, we design and implement an IDS for a particular context determined by a public dataset that includes the electrical power consumption of 21 devices in a smart house [57]. This process will provide insight into the previous modules and their adaptation to a particular scenario.

4.1 Context

The context of this case study is determined by the Almanac of Minutely Power dataset (AMPds) [57], which contains 524 544 registers collected from April 1st, 2012, to March 31st, 2013 (one year) and includes 11 measurements at one-minute intervals for 21 appliances that are being used in a house. As such, it could be seen as 21 smart power meters, each monitoring a different home device. Table 3 shows the list of devices monitored.

However, this public dataset does not provide sufficient information to fully apply our reference architecture as it lacks a local scenario with networks and elements such as those shown in Fig. 1. For this reason, we create a hypothetical scenario, like the one shown in Fig. 3, that would produce a dataset similar to AMPds. In our scenario, each power meter sends a message every minute to a central data acquisition server (Home Server in Fig. 3) with a timestamp and the values of 11 variables related to the electrical consumption of a device (e.g., timestamp, voltage, real power, reactive power, etc.). The acquisition server stores this data in a MySQL database which is accessible to the IDS.

The network-level module requires collecting and aggregating traffic from the power meter devices as it uses a pcap file as input. Thus, we have fabricated the network traffic that would have produced the dataset AMPds in our scenario, assuming that the MQTT application protocol was in place. For this, we have generated the sequence of packets according to the timestamp of each register in

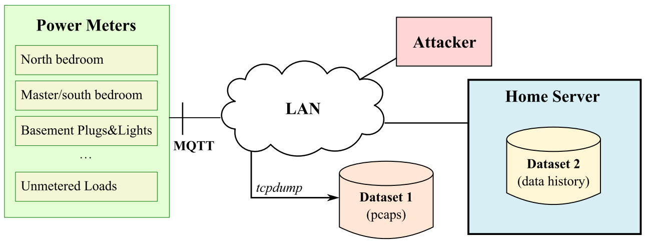

Fig. 3. Scenario in our case study.

Table 4
Testbed attacks.

ID	Sub-technique	Attack	Type	MITRE ID	Command/Description	Destination
AT1	TCP SYN flooding	DoS	Network	T1499.001	`hping3 -p <port> -S –flood <ip_target>`	Server
AT2	DDoS		Network	T1498.001	`hping3 -rand-source –d <data_size> <ip_target> -p <port> –flood`	Server
AT3	ICMP scan	Scanning	Network	T1595.001	`nmap -sO <ip>/<mask>`	LAN
AT4	ARP scan		Network	T1595	`nmap -sn <ip>/<mask>`	LAN
AT5	Service scan		Network	T1595.002	`nmap -sV <ip>`	Server
AT6	ARP Cache Poisoning	Sniffing	Network	T1557	`arpspoof -i <interface> -t <ip_target> <ip_gw>`	IoT device or router
AT7	Brute Force	Credential access	Network	T1110.001	`hydra -l <username> -P <path_to_wordlist> <IP> -t <number_of_threads> ssh`	Server
AT8	Command and Control	Remote access software	Network	T1219	`hping3 -S <ip_ext> -p <port_service>`	IoT device
AT9	False Data Injection	Data manipulation	Device	T1565	One variable from a device (voltage) has nonsense values (double than regular)	Server
AT10	False Command Injection	Data manipulation	Device	T1565	The device active power is set to always ON during 24 h	Server
AT11	False Command Injection	Data manipulation	Device	T1565	The device active power is randomly chosen between 0 and 2 P_{max}	Server
AT12	False Command Injection	Data manipulation	Device	T1565	The device is switched OFF during 24 h (E-DoS, false command...)	Server

the database.[4] The packets generated were collected in a pcap file with tcpdump. The tool used for this process is available in [58]. This module's first task is to generate and export IPFIX traffic flows every 5 min from the pcap file. We used Tranalyzer [59] for this task, activating the DPI feature. The pcap file and its associated IP flows for the first three months of data used in this paper are available in [58].

So, our IDS operates offline (rather than in real-time) using the previously mentioned datasets: AMPds as input for the application-level module, and the fabricated pcap as input for the network-level and signature-based modules.

4.2 Attacks considered

We have generated a set of attacks launched by an internal attacker to test the IDS performance. We believe internal adversaries can be a common case in smart homes when one IoT device or the Home Server has been compromised. Table 4 shows the list of attacks considered and their implementation method. The attacks can be classified according to the target and method as:

- Generic network attacks (AT1-8). These attacks have been carried out using the commands stated in Table 4 from a PC running Kali Linux. For testing, the packets produced with each attack have been captured and integrated in the corresponding pcap file (see above).
- Device-specific application-level attack (AT9-12). These attacks have been realized by data manipulation (i.e., changes in the data from the AMPds dataset). Additionally, AT11 was implemented eliminating the device network traffic.

The pcap files with the attacks generated can be downloaded from [58].

[4] To speed up the process, we escalated the timestamp value in the database so that the entire year could be produced in about 30 h.

Table 5
IoCs designed for network-level module.

ID	Description	Detector	Context
IoC1	Total ARP request	D1	G
IoC2	Total ARP response	D1	G
IoC3	Total different destination IP addresses	D2	G
IoC4	Total different TCP ports	D1	G
IoC5	Total different source IP addresses	D1	G
IoC6	Total number of flows	D1	G
IoC7	Total number of ssh flows (DPI)	D1	G
IoC8	Total ICMP flows	D1	G
IoC9	Total number of ARP response-total number of ARP requests	D1	G
IoC10	Time between flows initiated by a source IP address	D3	I

4.3 *Design of the anomaly-based IDS*

In this section, we carry out a design that follows the generic architecture from Section 3 and is geared toward detecting the list of attacks from Table 4.

4.3.1 *Network-level module: IoC design*

This module should detect network attacks (AT1-8) and the "power off" device attack (AT12) as they impact network traffic. This requires the definition of a set of indicators of compromise (IoC) that examine the properties of the traffic flows. Regarding this, some common attacks (e.g., DoS) could be detected by a sudden increase in network traffic flows. Others, such as scanning, could be related to the variability over time of ARP/ICMP packets or TCP ports. Given the periodic nature of the traffic in this particular scenario, it would also be interesting to monitor variations in the flows periodicity for each device, which could inform us about a device-off problem. The previous considerations, along with the definitions of attacks AT1-8 and AT12 (Table 4), led us to define the set of IoCs shown in Table 5. For each IoC, we describe the type of technique used to generate the alarm (see below).

As stated earlier, since flow reports are generated periodically (default is 5 min), the property examined by each IoC can be treated over time as a time-series over which we can detect outliers using well-known techniques [60]. The types of detectors used for this are:

- D1: Moving average with a sliding window of size W (10 in our experiments). An anomalous event will be triggered if the difference between the current value and the moving average is greater than K times the standard deviation of the entire series.
- D2: Similar to D1 but considering the absolute value of the last sample.
- D3: Exponential Weighted Moving Average (EWMA) and the standard deviation of the entire series of values. The parameters used are the threshold multiplier (K), the window used for the standard deviation (W) (the value used for α is derived from W as shown in (1)), and the constant to increase the threshold (C).

$$\alpha = 1 - \exp\left(-\ln(2)/W\right) \tag{1}$$

  An alarm will be produced if the current value is less than EWMA minus K times the standard deviation minus C; or if it is greater than EWMA plus K times the standard deviation plus C.

The context of IoC1-9 is global –Type G– (i.e., we aggregate the flow reports for all IP/mac) but IoC10 is defined for each IoT device individually –Type I–.

4.3.2 *Application-level module design*

This module is geared toward detecting abnormal data sent by each IoT device (i.e., power meter in our case study). Abnormal data may be caused by false command or data injection attacks such as AT9-11. For this task, we have designed two simple anomaly detectors that leverage the nature of the data produced by power meters:

1. **Detector of incoherent variable value: PCA+Q**. The rationale behind this detector is that the 11 variables included on each register are somewhat related. For example, a device's active power depends on the Voltage, current intensity, and power factor. Therefore, if one sensor sends an erroneous value (due to malfunction or malicious manipulation), such as in AT 9, it could be detected. The detection method is based on Q residuals over Principal Component Analysis (PCA) [61]. During the training phase, we find the maximum value of the sample residual (Q_{max}). In contrast, during the detection phase, we flag an anomaly if the residual found for that sample, $Q(s)$, holds the following condition:

$$Q(s) > K \cdot Q_{max}, \tag{2}$$

   where K is a parameter that controls the detection sensibility. The interested reader can find in Appendix A more details on how we implemented this detector.

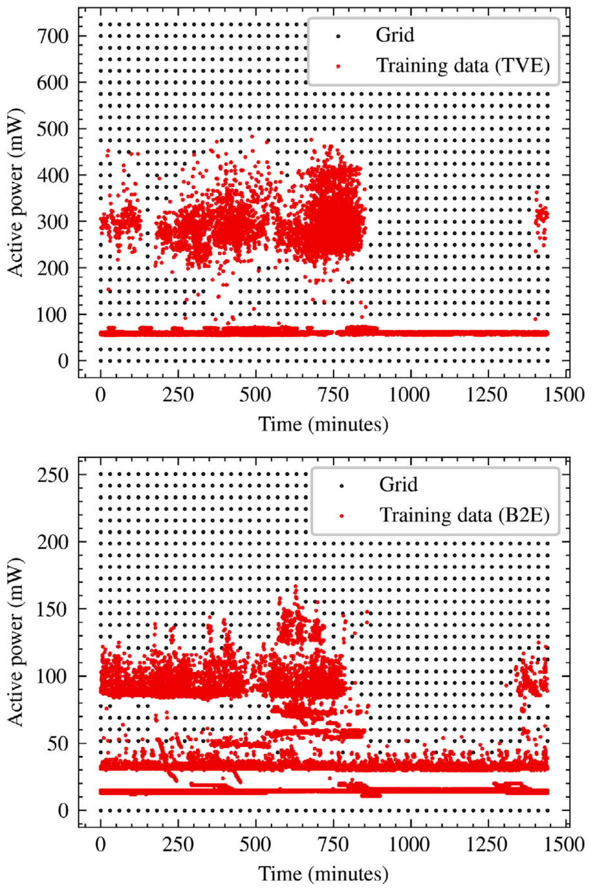

Fig. 4. Figures obtained after training for devices TV (top) and B2 (bottom).

2. **Detector of device's temporal mismatch: K-NN**. This detector is based on the assumption that, to some extent, a device's active power exhibits a similar daily pattern. This hypothesis does not necessarily holds for every device in the dataset. Still, it let us develop a simple but effective method to find power consumption anomalies in some IoT devices. Our technique is as follows. During the training phase, we examine the device's hourly active power over 24 h, collecting the values $(time, power)$ for that device as shown in Fig. 4 (red color). Obviously, the more frequent values show up as more dense areas: for example, if a device is always off between 1 and 4 am, the power values will always be 0 in that range. We aim to evaluate if a new sample $(time, power)$ is close enough to the behavior learned during the training phase using a binary classifier based on K-NN.
   To discriminate anomalies, we establish a collection of abnormal reference points that conform to a grid (uniformly spaced in time –steps of 30 min– and value –steps of 5% of the variable range). We suppress the grid points closer (according to K-NN) to the normal values learned during training to establish a clear separation between normal and abnormal points. Finally, during the detection phase, for every new sample $(time, power)$, we classify it as normal or not by measuring the euclidean distance to the K (3) nearest neighbors. If the three neighbors are points of the grid, we classify it as abnormal and, otherwise, normal.
   Fig. 4 shows the figure obtained after training for the devices TV (top) and B2 (bottom). It can be observed that sometimes it is easy to find a periodic pattern (e.g., TV is always off at certain hours of the day) which can be the case for some smart-home IoT devices. Other times, however, although less periodic, it can provide insight into common power patterns over time ranges such as is the case with B2.
   The two detectors designed in this module are suited to our particular dataset. There is a plethora of other anomaly detection techniques for detecting anomalies in the application-level data [53,62]. But, as stated in Section 1, the final techniques developed should ideally be adapted on each case to the particular dataset and attacks under consideration.

4.3.3 Other modules

The rest of the components of our system are:

- **External S-IDS**: we have used a popular signature-based IDS (Snort) to detect known attacks in the network traffic. We use Talos ruleset, activating the following pre-processors: sfportscan, arpspoof and ssh. The total number of rules in place were 626. More details are provided in Section 5.4.
- **Correlator**: this module aims at generating a reduced number of significant alerts. Observe that a single attack may generate events in various IoCs or be detected by the external S-IDS. However, we would like to have them aggregated into a single alert with sufficient information to validate the attack. In our implementation, we have used the Simple Event Correlator (SEC) engine [63]. The correlation rules infer the type of attack based on the events detected by the IoCs, application-level detectors or S-IDS. Thus, our approach to establish the correlation rules is to perform the attacks first, and infer the correlation rules *a posteriori* according to the results.

5 Performance evaluation

This section evaluates the system performance defined above against the attacks in our case study (see Table 4). In our experiments, we used the month of April in the dataset to train the models and the month of May for the validation (i.e., detection of attacks). We also carried out prospective tests with other months obtaining similar results (Appendix D shows the results obtained using June for validation rather than May), which suggests that these pieces of the dataset suffice to illustrate the system performance.

5.1 Performance indicators

We have used the following indicators:

- True Positive (TP): TP indicates an attack detected by at least one element (e.g., IoC). For testing the network-level module, we inserted attacks AT1-9 and AT12 in the input pcap file at random instants within the evaluation period, verifying that the timestamp of the event detected coincides with that of the attack inserted. For testing the application-level module, we followed a similar procedure. A TP is considered if an anomaly is seen in the poisoned sample introduced in the dataset.
- False Positive (FP): a FP happens when an abnormal event is detected, but an attack was not inserted into the original dataset.
- True Negative (TN): TN indicates normal sample classified as normal. It is calculated by subtracting the number of FPs to the total number of samples in the original dataset.
- False Negative (FN): FN indicates undetected attacks. It is calculated as the total number of malicious samples minus the total number of TP.

These four basic indicators allow one to evaluate the detection performance through various metrics such as True Positive Rate (TPR) and True Negative Rate (TNR), as presented in Eqs. (3) and (4), respectively.

$$TPR = TP/(TP + FN) \tag{3}$$

$$TNR = TN/(FP + TN) \tag{4}$$

However, since performance is compared in different scenarios, and the classes normal and anomalous are clearly unbalanced, we will use the geometric mean [64,65] as a performance indicator. This metric combines recall and specificity; hence, it is sensitive to detection capacity and false positives. Thus, for the remainder of this paper, the IDS performance metric will be given by Eq. (5).

$$\eta = \sqrt{TPR \cdot TNR} \tag{5}$$

5.2 Network-level module performance

To evaluate the TPR, we have carried out two tests for each attack type:

(a) Test 1: a single attack execution inserted randomly within the test period.
(b) Test 2: 100 attack executions inserted at random moments within the test period.

The second test was repeated 30 times and the results shown represent average values. Table 6 shows the results obtained for each IoC and test (Test 2 - Test 1). As expected, attacks AT1-8 have been detected, as has AT11, which is related to traffic from the device. Results show that the attack in Test 1 has always been detected. However, all the 100 attacks in Test 2 have not always been detected. This can be traced back to the fact that if two attack instances are too close in time, only the first one is detected as the detector based in MA and EWMA requires a guard period until it can be detected again, as shown in Fig. 5 UCL (Upper Control Limit). This is why the detection rate decreases from 100% to 86% in some IoCs.

We have also examined false positives (FP) in the events detected (i.e., when no attacks were introduced in the dataset). Only the IoC6 (number of flows) generated 21 FP. A closer look at these events revealed that they were always caused by anomalous data series (e.g., some IoT devices not transmitting their data). An example is shown in Fig. 5. It can be noticed that the number of flows (the black line in Fig. 5), which is usually constant over time, experience fluctuations (e.g., due to intermittent transmission of data between hours 2 and 5), following an abrupt reduction from 100 to 95 (because 5 devices have not transmitted data). When these

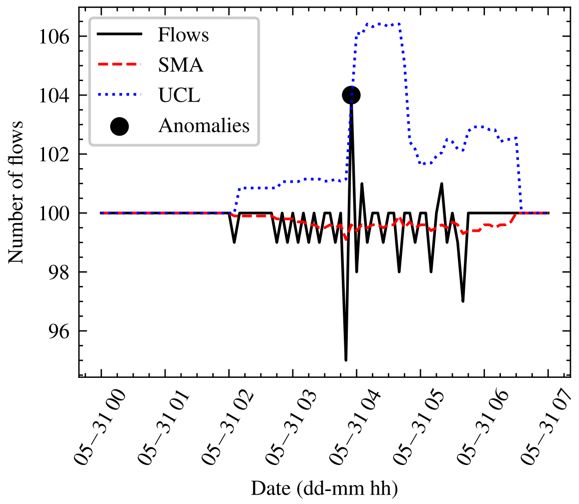

Fig. 5. IoC6 false positives (FP).

Table 6
Number of events detected by each IoC for each attack. Format: Test 2 (100 attacks) – Test 1 (one attack).

	IoC1	IoC2	IoC3	IoC4	IoC5	IoC6	IoC7	IoC8	IoC9	IoC10
AT1	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	**93 – 1**	0 – 0	0 – 0	0 – 0	**95 – 1**
AT2	0 – 0	0 – 0	0 – 0	0 – 0	**88 – 1**	**93 – 1**	0 – 0	0 – 0	0 – 0	0 – 0
AT3	0 – 0	0 – 0	**95 – 1**	**93 – 1**	0 – 0	**95 – 1**	0 – 0	**95 – 1**	0 – 0	**95 – 1**
AT4	**87 – 1**	**87 – 1**	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0
AT5	0 – 0	0 – 0	0 – 0	**90 – 1**	0 – 0	**90 – 1**	0 – 0	0 – 0	0 – 0	**92 – 1**
AT6	0 – 0	**86 – 1**	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	**86 – 1**	0 – 0
AT7	0 – 0	0 – 0	0 – 0	**92 – 1**	0 – 0	**92 – 1**	**92 – 1**	0 – 0	0 – 0	**96 – 1**
AT8	0 – 0	0 – 0	**92 – 1**	**92 – 1**	0 – 0	**40 – 1**	0 – 0	0 – 0	0 – 0	**100 – 1**
AT9	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0
AT10	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0
AT11	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0
AT12	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	0 – 0	**100 – 1**

devices resume their transmission another abrupt change from 95 to 104 flows follows, which triggers the detection of anomalous behavior. Thus, this FP is caused by disturbances in a time series that should be stable over time. As such, these disturbances are legitimately considered a rational sign of anomaly. This applies to the other 20 FP observed. In Fig. 5 we can also see that these anomalies also impact the limits of the control area (dotted blue line in Fig. 5), as it automatically adapts to changes, which in turn may cause that several consecutive attacks are detected as a single one.

At any rate, since the input dataset has 8640 samples (month of May), 21 false positive would provide a True Negative Rate of 99.76% for IoC6, which seems acceptable.

5.3 Application-level module performance

This module is composed of 2 detectors applied to all devices: one (PCA+Q) for detecting incoherent values in the variables within each sample (tested with AT9), and another one (K-NN), to detect anomalies in the daily pattern of each device's active power (tested with attacks AT10, AT11).

5.3.1 Detector PCA+Q (attack AT9)

To emulate attack AT9 we have modified the value of device's active power consumption at 100 random instants by replacing the original value with another one which can be up to a 50% greater or lower. In the PCA model, we have selected two components that explain 98% of the variance (see Appendix A). An event is triggered when a sample's Q residual is greater than the double of the maximum Q residual observed during training (i.e., $K = 2$). As an example, we show in Fig. 6 the Q residual obtained for the GR device in the test phase (i.e., including corrupted samples). In the plot, the red dotted line shows the limit (i.e., $2 \cdot Qmax\_train$) learned after training. It is visible that the corrupted samples exceed the Q limit, which enabled their detection. In our test, we evaluated the system before and after inserting the aforementioned 100 anomalies in order to obtain FP and TP respectively.

The performance (η_{AT9}) obtained for each device is summarized in Table 7. More details (e.g., TP, TN, FP, etc.) can be found in Appendix C. The overall (i.e., for all devices) average performance is 89%. It can be observed though, that the model did not work

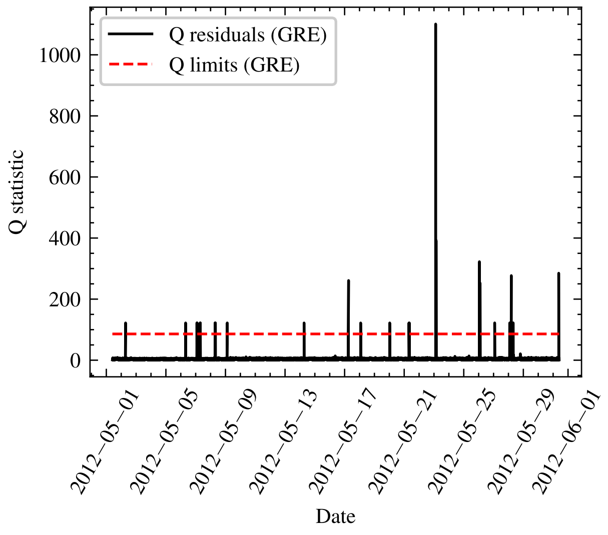

Fig. 6. Q statistics for device GRE.

Table 7
Data anomaly detection results summary.

Device	PCA+Q	K-NN	
	η_{AT9}	η_{AT10}	η_{AT11}
GR	1,00	0,99	0,94
WO	1,00	0,94	0,98
BM	1,00	0,94	0,89
UT	1,00	0,00	0,85
TV	1,00	0,98	0,86
B2	0,89	0,99	0,81
DW	1,00	0,74	0,93
B1	1,00	0,97	0,97
HT	1,00	0,98	0,80
FR	1,00	0,96	0,95
CD	1,00	1,00	0,97
OF	1,00	1,00	0,93
OU	0,00	0,56	0,53
EQ	1,00	0,39	0,85
WH	1,00	0,91	0,84
CW	1,00	1,00	0,94
DN	1,00	0,98	0,86
EB	0,00	1,00	0,96
FG	1,00	1,00	0,94
HP	1,00	1,00	1,00

for the electrical consumption of the outside plug (OU) and the electronics workbench (EB) as it did not detect anomalous values. This can be attributed to the presence of anomalous values in the training dataset that produced a very high value of Qmax. Another unexpected result was the number of FP obtained for B2 (see Appendix C). As with OU and EB, this is attributable to anomalous samples in the training datasets maybe due to malfunctioning of the device. Appendix C details some examples that justify this conclusion. If we excluded OU and EB, η_{AT9} would have been 99.4%, with a TPR of 100% . Therefore, we can conclude that the performance of this detector is satisfactory.

5.3.2 *Detector K-NN (attacks AT10, AT11)*

Our second detector is devoted to detects anomalous values on a device's active power based on the daily pattern observed during training. This technique is suited to detect attacks AT10 (a device is 24 h at maximum power) and AT11 (a device shows random power values at random instants).

The results obtained for each device are shown in Table 7 (η_{AT10} and η_{AT11}). It can be observed that, for AT10 (average η = 80%), the average True Positive Rate (TPR) was 80.9%, whereas TNR was 99.3% (see Appendix C). The results showed that the model seemed to be inadequate with three devices: OU, EQ and UT. After careful examination of the data associated to these devices, we observed that their active power consumption was very erratic and residual which prevented any predictive model to be of use. If we suppressed these devices from the performance indicators, we would have obtained a TPR of 93.1%, a TNR of 99.2% and the performance (η) would be 95.9%. In the case of AT11 attack, the TPR scored on average 79.58% while TNR is 99.5%, having an

Table 8
Snort results.

ID	Sub-technique	Total	Alarms for each signature ID							
		alarms	1	22	26	82	384	469	528	10 012 342
–	Clean traffic	0	0	0	0	0	0	0	0	0
AT1	TCP SYN flooding	0	0	0	0	0	0	0	0	0
AT2	DDoS	9	0	0	0	1	0	0	7	1
AT4	ICMP scan	47	0	0	1	0	23	23	0	0
AT5	ARP scan	0	0	0	0	0	0	0	0	0
AT6	Service scan	1	1	0	0	0	0	0	0	0
AT7	ARP Cache Poisoning	5	0	5	0	0	0	0	0	0
AT8	Brute Force	0	0	0	0	0	0	0	0	0
AT9	Command and Control	0	0	0	0	0	0	0	0	0

overall performance of 89%. Once again, we observe that the three aforementioned devices, OU, EQ, UT, performed worse than the rest. If we discarded these three devices, we would have obtained a performance of 91%.

5.4 External system: Snort

We have used the network traces (pcap) as input of Snort with the following configuration:

- Preprocessors activated: sfportscan, arpspoof, ssh
- Overall number of rules: 626 (377 detection rules, 153 decoder rules, 96 preprocessor rules)
- Snort's rules[5]: bad-traffic.rules, ddos.rules, dos.rules, exploit.rules, icmp-info.rules,icmp.rules, scan.rules, preprocessor.rules, decoder.rules

The results obtained are shown in Table 8 and show that Snort, with the configuration above, was able to detect 4 out of the 12 known network attacks (AT2, AT4, AT6, AT7) in the network traces.

5.5 Computational performance

Note that the IDS operates offline in our case study, as each module has been run standalone using the necessary input files. This section investigates the computational performance of the two main modules designed. This can be taken as a reference since the architecture of the SID (e.g., distributed or centralized) is to be decided. The computational performance of Snort and the correlation engine SEC (and other open-source alternatives) has been sufficiently studied in [67,68] respectively.

5.5.1 Network-level module

We have tested the computational performance of this module by measuring the use of CPU and memory during the processing of a month of traffic (input pcap size is 533 MB) in a PC with an AMD® A10-7860k processor and 16 GB of RAM. Fig. 7 shows the timeline of CPU usage during execution (100% represents a single CPU core). In black color, we plot the flow export process (including DPI) while the computation of IoCs is marked in red dashed line. It can be noted that more than 95% of CPU time was devoted to the flow exporting process. In our test, the use of RAM memory never exceeded 165 MB, which seems reasonable considering the actual hardware.

Our results show that our implementation processed 12 700 800 packets in less than 250 s. Thus obtaining a throughput of 50 803 packets per second.

5.5.2 Application-level module

Similarly, we have tested the computational performance of this module by analyzing the CPU and RAM occupation while processing 1 month of data on the same machine. Each sample has a size of 145B (i.e., 11 json-formatted fields). Fig. 8 illustrates the CPU usage for both detectors (PCA+Q and K-NN) executed sequentially. The results show that CPU usage is bounded between 50% and 400% (4 cores), while RAM demand is always below 162 MB. Furthermore, a month-length (907 200 samples) was processed in less than 60 s (15 120 samples per second).

The results obtained suggest that the network module implemented is more computationally demanding than the application-level one, mainly due to the flow export and DPI processes. Thus, these tasks should be distributed for scalability in challenging environments (i.e., several devices monitored). At any rate, the results obtained in our scenario suggest the feasibility of using our IDS online in a centralized architecture.

5.6 Discussion

In light of the previous results, we would like to elaborate on the following points.

[5] Available in [66].

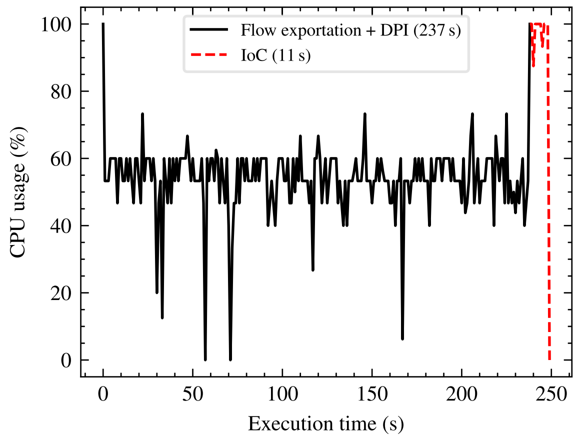

Fig. 7. Anomaly traffic detector CPU usage.

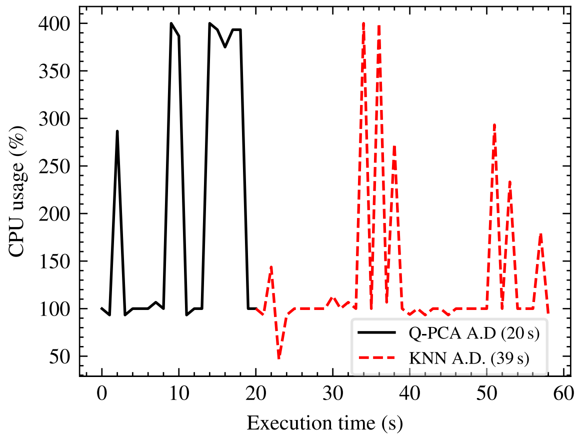

Fig. 8. Anomaly data detector CPU usage.

5.6.1 Detection sensibility

One can conclude that the overall performance of the IDS is acceptable for most devices and attacks tested in the case study ($\eta > 90\%$). The occasions where attacks passed unnoticed were due to abnormal values in the dataset. This does not indicate poor data quality but simply that, differently from the academic context, real-life operational datasets are likely to include unexpected data due, for instance, to device failures [69]. These unexpected phenomena may lead to undetected anomalies or false positives, temporarily undermining the detection technique used.

5.6.2 Event correlation

As a summary, Table 9 shows the detector element (i.e., indicator) from our IDS that have flagged each attack in our tests (considering the results from most of the devices). It can be observed that all attacks have been detected by at least one element. Thus, particularly in network attacks, the set of indicators activated for each attack could be seen as a fingerprint of the attack viewed by our IDS. As a consequence, Table 9 could be used to design the correlation directives used by the correlation engine. For example, a service scan attack (AT6) is detected by the events generated by IoC2 and IoC9. If both events are produced in a short time span, the correlation engine can generate a single alarm indicating the attack AT6 was caught by an anomaly. Moreover, because we also have an external signature-based IDS that has detected the same attack, we could send a single alarm indicating that attack AT6 has been seen by anomaly and also by signature, thus increasing the reliability of the detection. Although the correlation rules have not been included in this work for the sake of conciseness, we have implemented the correlation directives following the results in Table 9.

5.6.3 Comparison with other IDS

As stated in the Introduction, IDS should ideally be adapted to the target scenario. In this sense, none of the IDS used in Smart Home from Table 1 are potentially suited to detect the application-level anomalies that led to the detection of attacks AT9-12. In most cases, the application-level data is overlooked, so data manipulation attacks cannot be detected. Only Sikder et al. [40] consider

Table 9
Experiments summary.

	IoC1	IoC2	IoC3	IoC4	IoC5	IoC6	IoC7	IoC8	IoC9	IoC10	PCA+Q	K-NN	Snort
AT1						✓				✓			
AT2					✓	✓							✓
AT3			✓	✓		✓		✓		✓			
AT4	✓	✓											✓
AT5				✓		✓				✓			
AT6		✓							✓				✓
AT7				✓		✓	✓			✓			✓
AT8			✓	✓		✓				✓			
AT9											✓		
AT10												✓	
AT11												✓	
AT12										✓			

Table 10
PCA+Q results summary (AT9).

Device	TP	FP	TN	FN	η
GR	100	22	43 078	0	1,00
WO	100	0	43 100	0	1,00
BM	100	0	43 100	0	1,00
UT	100	0	43 100	0	1,00
TV	100	0	43 100	0	1,00
B2	100	8601	34 499	0	0,89
DW	100	0	43 100	0	1,00
B1	100	0	43 100	0	1,00
HT	100	0	43 100	0	1,00
FR	100	0	43 100	0	1,00
CD	100	0	43 100	0	1,00
OF	100	0	43 100	0	1,00
OU	0	5	43 095	100	0,00
EQ	100	0	43 100	0	1,00
WH	100	0	43 100	0	1,00
CW	100	0	43 100	0	1,00
DN	100	0	43 100	0	1,00
EB	0	0	43 100	100	0,00
FG	100	0	43 100	0	1,00
HP	100	0	43 100	0	1,00

application-level information in their system Aegis+, which observes the states of the connected smart home entities (sensors and devices) for different user activities and usage patterns and builds a contextual model to differentiate between malicious and benign behavior. In this case, however, network traffic is disregarded (AT1-9). Furthermore, their model is based on states which is not fit to detect the application-level attacks implemented in this work.

Finally, notice that detecting attacks based only on a standard signature-based IDS such as Snort would not have ultimately detected all network attacks AT1-9 with the standard ruleset used in our case study.

6 Limitations

Our work rests on some assumptions, and its generalization is constrained by limitations that should be forewarned.

Architecture limitations. We believe that the main limitations and challenges of the architecture introduced in Section 3 are the following:

- A single point must receive the traffic flows from all devices. Besides being a single point of failure (exposed to DoS attacks), this requires gathering the network traffic from all smart devices, which can be challenging due to the network design (e.g., direct communication between two devices on the same subnetwork might not be collected), or overload the network devices (e.g., switch) which in turn are required to feature port mirroring or directly generate the traffic flows reports. These features are not always present in domestic-level switches or routers. To improve scalability, the traffic collection and flow generation could be distributed or use dedicated devices. The protection of the IDS against cyberattacks should also be considered.
- Application data collection. Gathering the application-level dialogues of heterogeneous devices can be challenging if a Home Server is not in place or its database is not accessible. Some smart devices send messages (commonly encrypted) to their respective vendor servers hosted on the Internet. Using the device manufacturer API to fetch state history could be tentatively considered in such a case. Clearly, the application-level module can only be applied to smart devices from which data is available.

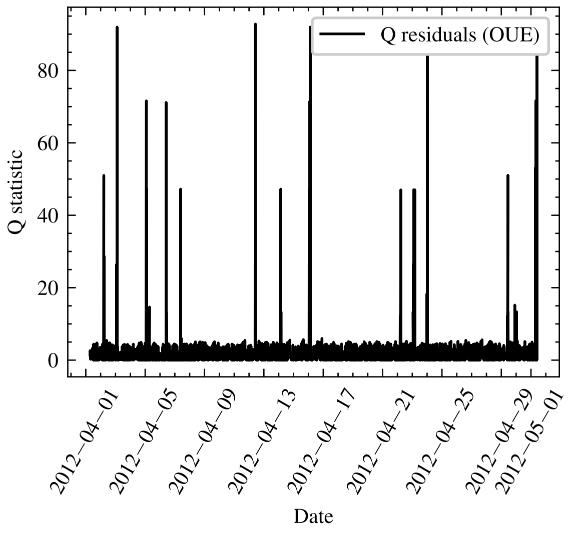

Fig. 9. Q statistics for device OU (first training phase).

- Undetected attacks. The network module cannot detect attacks that do not leave trace in the traffic pattern (e.g., APT). Similarly, detecting anomalies in the application-level data can be challenging due to the need for quality device-specific datasets and re-training [64].

IDS and case study limitations. The IDS developed in our case study (Section 4) also exhibits some limitations besides those inherent to the architecture:

- We have defined a set of detectors (IoCs in Table 4) suited to the generic network attacks listed and implemented as shown in Table 3. Different network attacks or techniques have not been tested. Moreover, we have not collected real traffic in our scenario but generated synthetic one from the measurements sent by the 21 smart power meters from the dataset. So, the generalization of our results to other real-life scenarios is limited by these facts.
- Our scenario is determined by the public dataset that includes 21 electric power meters connected to different types of smart devices. In this sense, our application-level data relates to smart devices' electrical consumption, not functionality. This scenario has certain advantages. For instance, the two data anomaly detectors developed are generic and applicable to any smart device that plugs into an electric outlet despite its functionality. Thus, attacks that leave a trace in the electric consumption temporal pattern can potentially be detected. But it also has limitations. For example, a device failure may be interpreted as an attack or the inability to detect attacks that do not impact a smart device's power consumption temporal pattern. Therefore, it seems advisable to complement the two classifiers defined in our work with other device-specific anomaly detection techniques-

The previous limitations also pave the way for future research.

7 Conclusions and further work

We have designed a generic architecture to detect attacks in the context of a Smart Home. Our approach is purely passive, flexible, modular, and efficient. We used one module to detect network attacks through IoCs that can be computed using traffic flow reports. Another module has been devoted to detecting anomalies in the data exchanged by IoT applications. The latter requires the design of specialized detectors on a per-case basis. External components, such as SIDS, can also be added. We have particularized our architecture for a case use that includes common threats in the smart-home literature, defining ten simple but efficient IoCs and two application-level detectors. We have used a public dataset to test the performance of our system, obtaining a detection capacity greater than 90% and an acceptable rate of false positives for most devices, being able to detect all attacks tested.

Our architecture can be adapted to suit different scenarios and attacks by including or defining new detectors, as suggested in the literature. It can also be a reference for researchers to add new IoCs or detection modules thanks to the public dataset used.

We can identify the following potential research lines for further work.

- Data privacy. In this work, we have assumed a household context (i.e., no third party is involved in the IDS operation). However, a different context may involve a third-party handling user data (e.g., a distributed cloud-based IDS using agents installed in smart homes). In such a case, data privacy and security should be a concern, and communications should be protected with lightweight cryptographic techniques such as in [70,71] to ensure the information received by each module

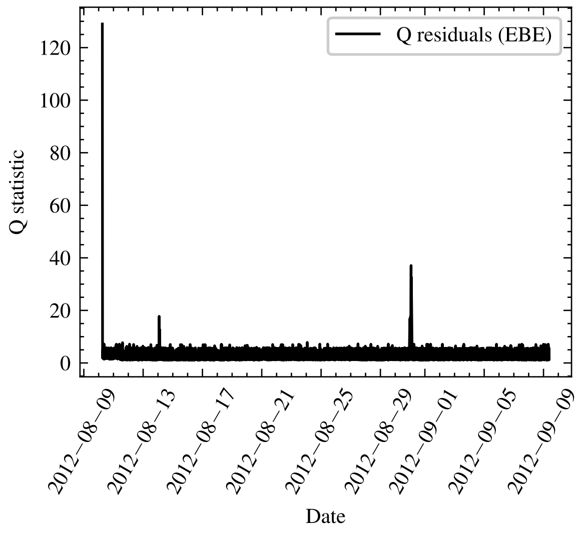

Fig. 10. Q statistics for device EB (first training phase).

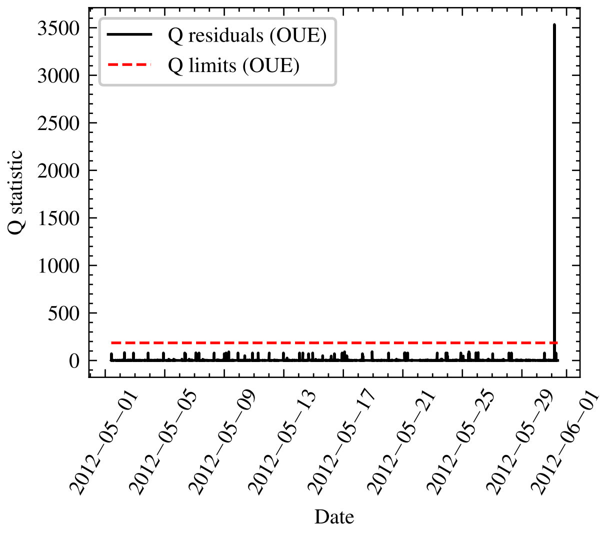

Fig. 11. Q statistics for device OU (second training phase).

and the alarms generated by the correlator are protected and confidential. Content security and privacy in a highly distributed approach can also benefit from innovative approaches such as Named Data Networking [72] or trajectory mining [73].

- Exploring the bounds of attack detection through anomalies in the electric consumption of smart devices. It would be interesting to research the circumstances under which attacks on smart devices (according to their type) are traceable through the electrical consumption footprint of the device [74,75].
- New application-specific anomaly detection techniques that extended the range of detectable attacks to those specific to the smart device functionality that could complement the detectors developed in this work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All datasets and tools to reproduce our work are shared in a public repository, which is referred in the Manuscript

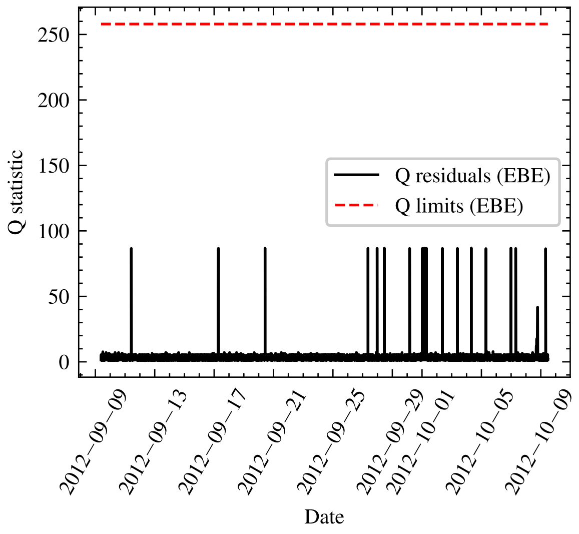

Fig. 12. Q statistics for device EB (second training phase).

Acknowledgments

This work has been partly funded by the research grant PID2020-115199RB-I00 provided by the Spanish ministry of Industry under the contract MICIN/AEI/10.13039/501100011033, and also by FEDER/Junta de Andalucía-Consejería de Transformación Económica, Industria, Conocimiento y Universidades under projects PYC20-RE-087-USE and A-TIC-224-UGR20.

Appendix A. Residual control method

An detailed explanation of the procedure for residual control can be found in [61]. A method works as follows:

- Given the training data matrix X with n observations and m correlated variables, we first normalize data to obtain a mean and variance of 0 and 1 respectively. Then, we perform a singular value decomposition of the matrix X which yields a new matrix with the principal components (which in turn are a lineal combination of the variables). Thus, we can express the training data matrix as $X = T \cdot P^T$, where T is the score matrix, and P is the loading matrix, whose columns are eigenvectors associated with X covariance matrix, $\Sigma = \frac{1}{n-1} \cdot X^T \cdot X = P \Lambda P^T$, and Λ is a diagonal matrix composed by the eigenvalues incrementally sorted as $(\lambda_1, \lambda_2, \ldots.)$.
- When Principal Component Analysis is applied to reduce dimensionality, we only select the first A principal components of the matrix P, in the spirit that these A components describe a significant part of the covariance observed in the data. Thus, we keep P_A, matrix whose dimension is mxA. During the test phase, whenever we have a new sample with m variables x_n, we calculate the vector of scores as $t_n = x_n \cdot P_A$.
- Given that we have reduced the number of components, the estimation P_A will exhibit some differences with respect to the original sample. Indeed, the error between the sample and its estimate is $e_n = x_n - -t_n \cdot P_A^T$. In general terms, the residual matrix indicates the error that is produced after reducing the number of components with respect to the original sample. The statistics Q can be defined as $Q = || (I_m - P_A P_A^T) ||^2$, and indicates the projection of a sample in the subspace of residuals, providing a generic indicator of how well a sample suits to the PCA model.
- We use a simple method for residual control. We take those components that justify 95% of the variance. During the training phase, we store the greatest residual value achieved Q_{max}. During the test phase, we calculate the residual for each sample and classify it as anomalous if it is K times greater than Q_{max},

Appendix B. OU and EB residual control discussion for PCA+Q

It can be observed that our model does not detect anomalous values for devices OUE y EBE. Examining the values of Q residuals obtained during the training stage, we observe unexpectedly high Qmax values for OU (Fig. 9) and EB (Fig. 10) of 90 y 120 respectively.

This is attributable to anomalous values in the dataset for these devices. Because Qmax observed during training was so high, we never obtain values greater than 2 times Qmax in the contaminated samples. This can be observed in Fig. 11 and Fig. 12 (the red line represents 2Qmax, and the blue line represents the observed Q values during test).

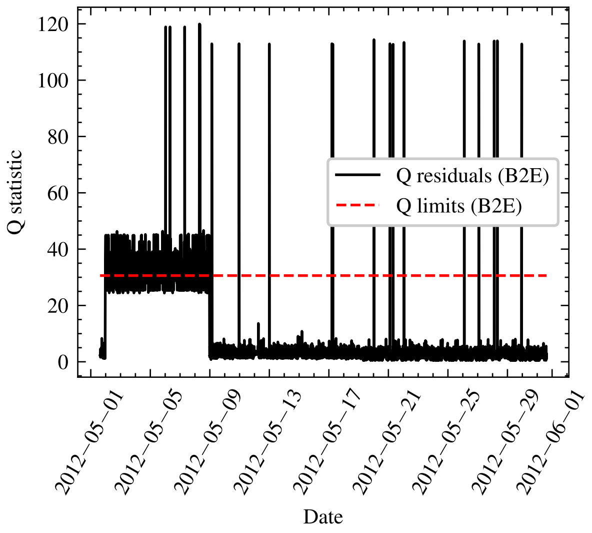

Fig. 13. Q statistics for device B2E.

Table 11
K-NN results for AT10.

Device	TP	FP	TN	FN	TPR	TNR	η
GR	1423	34	41 726	17	0,99	1,00	0,99
WO	1260	0	41 760	180	0,88	1,00	0,94
BM	1349	2743	39 017	91	0,94	0,93	0,94
UT	0	1	41 759	1440	0,00	1,00	0,00
TV	1383	382	41 378	57	0,96	0,99	0,98
B2	1406	241	41 519	34	0,98	0,99	0,99
DW	792	128	41 632	648	0,55	1,00	0,74
B1	1356	78	41 682	84	0,94	1,00	0,97
HT	1379	177	41 583	61	0,96	1,00	0,98
FR	1321	56	41 704	119	0,92	1,00	0,96
CD	1440	117	41 643	0	1,00	1,00	1,00
OF	1440	92	41 668	0	1,00	1,00	1,00
OU	452	48	41 712	988	0,31	1,00	0,56
EQ	222	2	41 758	1218	0,15	1,00	0,39
WH	1199	522	41 238	241	0,83	0,99	0,91
CW	1440	120	41 640	0	1,00	1,00	1,00
DN	1396	335	41 425	44	0,97	0,99	0,98
FG	1440	116	41 644	0	1,00	1,00	1,00
HP	1440	75	41 685	0	1,00	1,00	1,00

Finally, it can be noticed an elevated number of FP obtained in B2. In Fig. 13 we show the residuals obtained during the training stage.

It can be noticed that, besides the peaks attributable to contaminated samples, several samples exhibit anomalous values, probably due to sensor malfunction. As such, these 8601 FP should theoretically be considered as TP since they detect anomalous values already in the dataset.

Appendix C. Data anomaly detectors results

This appendix presents the results obtained for some of the presented Data Anomaly Detectors. Table 10 provides a summary of the results obtained for the PCA+Q detector for each device, indicating the number of TP, FP, TN, FN, and the performance indicator η.

Similarly, Table 11 summarizes the results obtained for the K-NN detector for AT10, also including the TNR and TPR values.

Finally, Table 12 summarizes the results obtained for the K-NN detector for AT11.

Appendix D. Results obtained using june for validation

As stated in Section 5, system performance was also tested with different months (rather than May) to confirm that similar results can be obtained. In this appendix, we repeat the same experiments in Section 5, but using June data for the tests. To do so, the same performance indicators (TP, FP, etc.) have been considered.

Table 12
K-NN results for AT11.

Device	TP	FP	TN	FN	TPR	TNR	η
GR	1265	31	41 729	175	0,88	1,00	0,94
WO	1390	0	41 760	50	0,97	1,00	0,98
BM	1212	2496	39 264	228	0,84	0,94	0,89
UT	1038	1	41 759	402	0,72	1,00	0,85
TV	1078	272	41 488	362	0,75	0,99	0,86
B2	951	155	41 605	489	0,66	1,00	0,81
DW	1252	108	41 652	188	0,87	1,00	0,93
B1	1359	68	41 692	81	0,94	1,00	0,97
HT	919	139	41 621	521	0,64	1,00	0,80
FR	1291	54	41 706	149	0,90	1,00	0,95
CD	1369	76	41 684	71	0,95	1,00	0,97
OF	1243	79	41 681	197	0,86	1,00	0,93
OU	410	47	41 713	1030	0,28	1,00	0,53
EQ	1036	0	41 760	404	0,72	1,00	0,85
WH	1017	395	41 365	423	0,71	0,99	0,84
CW	1272	82	41 678	168	0,88	1,00	0,94
DN	1073	207	41 553	367	0,75	1,00	0,86
FG	1322	94	41 666	118	0,92	1,00	0,96
HP	1276	71	41 689	164	0,89	1,00	0,94

Table 13
Number of events detected by each IoC (out of 100) with alternative validation data (June).

	IoC1	IoC2	IoC3	IoC4	IoC5	IoC6	IoC7	IoC8	IoC9	IoC10
AT1	0	0	0	0	0	**92**	0	0	0	**92**
AT2	0	0	0	0	**90**	**93**	0	0	0	0
AT3	0	0	**90**	**88**	0	**90**	0	**90**	0	**95**
AT4	**89**	**89**	0	0	0	0	0	0	0	0
AT5	0	0	0	**88**	0	**89**	0	0	0	**92**
AT6	0	**90**	0	0	0	0	0	0	**90**	0
AT7	0	0	0	**89**	0	**90**	**90**	0	0	**96**
AT8	0	0	**89**	**98**	0	**50**	0	0	0	**100**
AT9	0	0	0	0	0	0	0	0	0	0
AT10	0	0	0	0	0	0	0	0	0	0
AT11	0	0	0	0	0	0	0	0	0	0
AT12	0	0	0	0	0	0	0	0	0	**100**

Table 14
Results obtained with alternative validation data (June).

(a) AT9 results (PCA+Q)

Device	TP	FP	TN	FN	η
GR	100	0	43 096	0	1,00
WO	100	0	43 096	0	1,00
BM	100	0	43 096	0	1,00
UT	100	0	43 096	0	1,00
TV	100	0	43 096	0	1,00
B2	100	11 074	32 022	0	0,86
DW	100	0	43 096	0	1,00
B1	100	1	43 095	0	1,00
HT	100	0	43 096	0	1,00
FR	100	0	43 096	0	1,00
CD	100	0	43 096	0	1,00
OF	100	0	43 096	0	1,00
OU	0	0	43 096	100	0,00
EQ	100	0	43 096	0	1,00
WH	100	0	43 096	0	1,00
CW	100	0	43 096	0	1,00
DN	100	0	43 096	0	1,00
EB	0	0	43 096	100	0,00
FG	100	0	43 096	0	1,00
HP	100	0	43 096	0	1,00

(b) AT10 results (KNN)

Device	TP	FP	TN	FN	TPR	TNR	η
GR	1423	1	41 755	17	0,99	1,00	0,99
WO	1260	12	41 744	180	0,88	1,00	0,94
BM	1349	450	41 306	91	0,94	0,99	0,96
UT	0	0	41 756	1440	0,00	1,00	0,00
TV	1383	369	41 387	57	0,96	0,99	0,98
B2	1406	166	41 590	34	0,98	1,00	0,99
DW	792	45	41 711	648	0,55	1,00	0,74
B1	738	5	42 453	0	1,00	1,00	1,00
HT	1379	474	41 282	61	0,96	0,99	0,97
FR	1321	0	41 756	119	0,92	1,00	0,96
CD	1440	151	41 605	0	1,00	1,00	1,00
OF	1440	14	41 742	0	1,00	1,00	1,00
OU	452	133	41 623	988	0,31	1,00	0,56
EQ	222	3	41 753	1218	0,15	1,00	0,39
WH	1199	755	41 001	241	0,83	0,98	0,90
CW	592	251	42 353	0	1,00	0,99	1,00
DN	1396	257	41 499	44	0,97	0,99	0,98
FG	1440	144	41 612	0	1,00	1,00	1,00
HP	1440	68	41 688	0	1,00	1,00	1,00

(c) AT11 results (KNN)

Device	TP	FP	TN	FN	TPR	TNR	η
GR	1111	1	41 755	329	0,77	1,00	0,88
WO	1351	12	41 744	89	0,94	1,00	0,97
BM	1016	450	41 306	424	0,71	0,99	0,84
UT	911	0	41 756	529	0,63	1,00	0,80
TV	798	379	41 377	642	0,55	0,99	0,74
B2	608	166	41 590	832	0,42	1,00	0,65
DW	1104	46	41 710	336	0,77	1,00	0,88
B1	1309	5	41 751	131	0,91	1,00	0,95
HT	607	464	41 292	833	0,42	0,99	0,65
FR	1159	0	41 756	281	0,80	1,00	0,90
CD	1324	255	41 501	116	0,92	0,99	0,96
OF	1102	14	41 742	338	0,77	1,00	0,87
OU	172	128	41 628	1268	0,12	1,00	0,35
EQ	4	3	41 753	1436	0,00	1,00	0,05
WH	612	740	41 016	828	0,43	0,98	0,65
CW	1170	241	41 515	270	0,81	0,99	0,90
DN	827	257	41 499	613	0,57	0,99	0,76
FG	1222	146	41 610	218	0,85	1,00	0,92
HP	1148	68	41 688	292	0,80	1,00	0,89

First, Table 13 represents the number of events detected by each compromise indicator (implemented in the network module), when randomly introducing 100 attack executions into the test period. As expected, a significant percentage of events have been detected for AT1-8 and AT11, obtaining similar results to those obtained in Section 5. In fact, the detection rate is at least as good as when May was used for testing.

Then, the application-level module was repeated, taking data from June. Table 14a, b and c, summarize the performance results obtained for AT9 (PCA+Q), AT10 (K-NN), and AT11 (K-NN), respectively. Again, a similar performance is obtained when compared to the results in Appendix C.

References

[1] P.K. Malik, R. Sharma, R. Singh, A. Gehlot, S.C. Satapathy, W.S. Alnumay, D. Pelusi, U. Ghosh, J. Nayak, Industrial internet of things and its applications in industry 4.0: State of the art, Comput. Commun. 166 (2021) 125–139.

[2] J. Jin, J. Gubbi, S. Marusic, M. Palaniswami, An information framework for creating a smart city through internet of things, IEEE Internet Things J. 1 (2) (2014) 112–121.

[3] Y. Jie, J.Y. Pei, L. Jun, G. Yun, X. Wei, Smart home system based on iot technologies, in: 2013 International Conference on Computational and Information Sciences, IEEE, 2013, pp. 1789–1791.

[4] W.H. Hassan, et al., Current research on internet of things (IoT) security: A survey, Comput. Netw. 148 (2019) 283–294.

[5] R. Roman, J. Zhou, J. Lopez, On the features and challenges of security and privacy in distributed internet of things, Comput. Netw. 57 (10) (2013) 2266–2279.

[6] A.S. Syed, D. Sierra-Sosa, A. Kumar, A. Elmaghraby, IoT in smart cities: a survey of technologies, practices and challenges, Smart Cities 4 (2) (2021) 429–475.

[7] T.A. Abdullah, W. Ali, S. Malebary, A.A. Ahmed, A review of cyber security challenges attacks and solutions for internet of things based smart home, Int. J. Comput. Sci. Netw. Secur. 19 (9) (2019) 139.

[8] Smart Home Statistics website, 2022, Accessed: 2022-09-30, https://comfyliving.net/smart-home-statistics/.

[9] R. Heartfield, G. Loukas, S. Budimir, A. Bezemskij, J.R. Fontaine, A. Filippoupolitis, E. Roesch, A taxonomy of cyber-physical threats and impact in the smart home, Comput. Secur. 78 (2018) 398–428.

[10] J. Qi, A. Hahn, X. Lu, J. Wang, C.-C. Liu, Cybersecurity for distributed energy resources and smart inverters, IET Cyber-Phys. Syst. Theor. Appl. 1 (1) (2016) 28–39.

[11] J. Johnson, Roadmap for Photovoltaic Cyber Security, Sandia National Laboratories, 2017.

[12] J. Ye, A. Giani, A. Elasser, S.K. Mazumder, C. Farnell, H.A. Mantooth, T. Kim, J. Liu, B. Chen, G.-S. Seo, et al., A review of cyber–physical security for photovoltaic systems, IEEE J. Emerg. Sel. Top. Power Electron. 10 (4) (2021) 4879–4901.

[13] N. Komninos, E. Philippou, A. Pitsillides, Survey in smart grid and smart home security: Issues, challenges and countermeasures, IEEE Commun. Surv. Tutor. 16 (4) (2014) 1933–1954.

[14] Nist, E. Aroms, NIST Special Publication 800-94 Guide To Intrusion Detection and Prevention Systems (IDPS), CreateSpace, Scotts Valley, CA, 2012.

[15] K. Ghirardello, C. Maple, D. Ng, P. Kearney, Cyber security of smart homes: Development of a reference architecture for attack surface analysis, in: Living in the Internet of Things: Cybersecurity of the IoT-2018, IET, 2018, pp. 1–10.

[16] M. Mazzara, I. Afanasyev, S.R. Sarangi, S. Distefano, V. Kumar, M. Ahmad, A reference architecture for smart and software-defined buildings, in: 2019 IEEE International Conference on Smart Computing, SMARTCOMP, IEEE, 2019, pp. 167–172.

[17] W. Ali, G. Dustgeer, M. Awais, M.A. Shah, IoT based smart home: Security challenges, security requirements and solutions, in: 2017 23rd International Conference on Automation and Computing, ICAC, IEEE, 2017, pp. 1–6.

[18] J.F. DeFranco, M. Kassab, Smart home research themes: An analysis and taxonomy, Procedia Comput. Sci. 185 (2021) 91–100.

[19] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, Survey of intrusion detection systems: techniques, datasets and challenges, Cybersecurity 2 (1) (2019) 1–22.

[20] B.B. Zarpelão, R.S. Miani, C.T. Kawakani, S.C. de Alvarenga, A survey of intrusion detection in internet of things, J. Netw. Comput. Appl. 84 (2017) 25–37.

[21] M.F. Elrawy, A.I. Awad, H.F. Hamed, Intrusion detection systems for IoT-based smart environments: a survey, J. Cloud Comput. 7 (1) (2018) 1–20.

[22] A. Khraisat, A. Alazab, A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges, Cybersecurity 4 (2021) 1–27.

[23] C. Kolias, A. Stavrou, J. Voas, I. Bojanova, R. Kuhn, Learning internet-of-things security" hands-on", IEEE Secur. Priv. 14 (1) (2016) 37–46.

[24] B. Tushir, Y. Dalal, B. Dezfouli, Y. Liu, A quantitative study of ddos and e-ddos attacks on wifi smart home devices, IEEE Internet Things J. 8 (8) (2020) 6282–6292.

[25] A. Sivanathan, F. Loi, H.H. Gharakheili, V. Sivaraman, Experimental evaluation of cybersecurity threats to the smart-home, in: 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS, IEEE, 2017, pp. 1–6.

[26] E. Anthi, L. Williams, A. Javed, P. Burnap, Hardening machine learning denial of service (DoS) defences against adversarial attacks in IoT smart home networks, Comput. Secur. 108 (2021) 102352.

[27] Y. Alshboul, A.A.R. Bsoul, M. Al Zamil, S. Samarah, Cybersecurity of smart home systems: Sensor identity protection, J. Netw. Syst. Manage. 29 (3) (2021) 1–27.

[28] T.M. McGee, Evaluating the cyber security in the internet of things: Smart home vulnerabilities (Ph.D. thesis), Colorado Technical University, 2016.

[29] M.N. Anwar, M. Nazir, K. Mustafa, Security threats taxonomy: Smart-home perspective, in: 2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA)(Fall), IEEE, 2017, pp. 1–4.

[30] K. Karimi, S. Krit, Smart home-smartphone systems: Threats, security requirements and open research challenges, in: 2019 International Conference of Computer Science and Renewable Energies, ICCSRE, IEEE, 2019, pp. 1–5.

[31] J.C. Sapalo Sicato, P.K. Sharma, V. Loia, J.H. Park, Vpnfilter malware analysis on cyber threat in smart home network, Appl. Sci. 9 (13) (2019) 2763.

[32] A. Sajeev, H.-S. Rajamani, Cyber-attacks on smart home energy management systems under aggregators, in: 2020 International Conference on Communications, Computing, Cybersecurity, and Informatics, CCCI, IEEE, 2020, pp. 1–5.

[33] L. Huraj, M. Šimon, T. Horák, Resistance of IoT sensors against DDoS attack in smart home environment, Sensors 20 (18) (2020) 5298.

[34] J. Bhayo, S. Hameed, S.A. Shah, An efficient counter-based ddos attack detection framework leveraging software defined iot (sd-iot), IEEE Access 8 (2020) 221612–221631.

[35] A. Arabo, Cyber security challenges within the connected home ecosystem futures, Procedia Comput. Sci. 61 (2015) 227–232.

[36] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, X. Bellekens, A taxonomy and survey of intrusion detection system design techniques, network threats and datasets, 2018.

[37] M.F. Umer, M. Sher, Y. Bi, Flow-based intrusion detection: Techniques and challenges, Comput. Secur. 70 (2017) 238–254.

[38] F. Alsakran, G. Bendiab, S. Shiaeles, N. Kolokotronis, Intrusion detection systems for smart home iot devices: experimental comparison study, in: International Symposium on Security in Computing and Communication, Springer, 2019, pp. 87–98.

[39] P. Nespoli, D. Díaz-López, F.G. Mármol, Cyberprotection in IoT environments: A dynamic rule-based solution to defend smart devices, J. Inf. Secur. Appl. 60 (2021) 102878.

[40] A.K. Sikder, L. Babun, A.S. Uluagac, Aegis+ a context-aware platform-independent security framework for smart home systems, Digit. Threat. Res. Pract. 2 (1) (2021) 1–33.

[41] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, P. Burnap, A supervised intrusion detection system for smart home IoT devices, IEEE Internet Things J. 6 (5) (2019) 9042–9053.

[42] O. Brun, Y. Yin, E. Gelenbe, Y.M. Kadioglu, J. Augusto-Gonzalez, M. Ramos, Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments, in: International ISCIS Security Workshop, Springer, Cham, 2018, pp. 79–89.

[43] M. Gajewski, J. Mongay Batalla, G. Mastorakis, C.X. Mavromoustakis, Anomaly traffic detection and correlation in smart home automation IoT systems, Trans. Emerg. Telecommun. Technol. (2020) e4053.

[44] A. Nicheporuk, A. Nicheporuk, A. Sachenko, O. Sachenko, A. Kazantsev, A system for detecting anomalies and identifying smart home devices using collective communication., in: IntelITSIS, 2021, pp. 386–397.

[45] T. Li, Z. Hong, L. Yu, Machine learning-based intrusion detection for iot devices in smart home, in: 2020 IEEE 16th International Conference on Control & Automation, ICCA, IEEE, 2020, pp. 277–282.

[46] P. Illy, G. Kaddoum, K. Kaur, S. Garg, ML-based IDPS enhancement with complementary features for home IoT networks, IEEE Trans. Netw. Serv. Manag. (2022).

[47] J. White, P. Legg, Unsupervised one-class learning for anomaly detection on home IoT network devices, in: 2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), IEEE, 2021, pp. 1–8.

[48] M. Nobakht, V. Sivaraman, R. Boreli, A host-based intrusion detection and mitigation framework for smart home IoT using OpenFlow, in: 2016 11th International Conference on Availability, Reliability and Security, ARES, IEEE, 2016, pp. 147–156.

[49] N. Vakakis, O. Nikolis, D. Ioannidis, K. Votis, D. Tzovaras, Cybersecurity in SMEs: The smart-home/office use case, in: 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD, IEEE, 2019, pp. 1–7.

[50] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, A. Wahab, A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions, Electronics 9 (7) (2020) 1177.

[51] A. Thakkar, R. Lohiya, A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges, Arch. Comput. Methods Eng. 28 (2021) 3211–3243.

[52] P. Aitken, B. Claise, B. Trammell, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information, RFC 7011, 2013, http://dx.doi.org/10.17487/RFC7011, URL https://www.rfc-editor.org/info/rfc7011.

[53] N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: A comprehensive survey, J. Netw. Comput. Appl. 128 (2019) 33–55.

[54] A. Gaddam, T. Wilkin, M. Angelova, J. Gaddam, Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions, Electronics 9 (3) (2020) 511.

[55] F. Alsakran, G. Bendiab, S. Shiaeles, N. Kolokotronis, Intrusion detection systems for smart home IoT devices: experimental comparison study, in: Security in Computing and Communications: 7th International Symposium, SSCC 2019, Trivandrum, India, December 18–21, 2019, Revised Selected Papers, Springer, 2020, pp. 87–98.

[56] A. Müller, C. Göldi, B. Tellenbach, B. Plattner, S. Lampart, Event correlation engine, (Master's Thesis), Department of Information Technology and Electrical Engineering, 2009, Eidgenössische Technische Hochschule Zürich.

[57] S. Makonin, F. Popowich, L. Bartram, B. Gill, I.V. Bajić, Ampds: A public dataset for load disaggregation and eco-feedback research, in: 2013 IEEE Electrical Power & Energy Conference, IEEE, 2013, pp. 1–6.

[58] V. Mayor, A. Lara, R. Estepa, A. Estepa, DIT - public / ids smart home supplemental material · gitlab, 2022, URL https://gitlab.com/dit-public/ids-smart-home-supplemental-material.

[59] S. Burschka, B. Dupasquier, Tranalyzer: Versatile high performance network traffic analyser, in: 2016 IEEE Symposium Series on Computational Intelligence, SSCI, IEEE, 2016, pp. 1–8.

[60] N. Laptev, S. Amizadeh, I. Flint, Generic and scalable framework for automated time-series anomaly detection, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1939–1947.

[61] F. Harrou, F. Kadri, S. Chaabane, C. Tahon, Y. Sun, Improved principal component analysis for anomaly detection: Application to an emergency department, Comput. Ind. Eng. (ISSN: 0360-8352) 88 (2015) 63–77, http://dx.doi.org/10.1016/j.cie.2015.06.020, URL https://www.sciencedirect.com/science/article/pii/S036083521500279X.

[62] M. Ahmed, A.N. Mahmood, J. Hu, A survey of network anomaly detection techniques, J. Netw. Comput. Appl. 60 (2016) 19–31.

[63] R. Vaarandi, B. Blumbergs, E. Çalışkan, Simple event correlator-best practices for creating scalable configurations, in: 2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision, IEEE, 2015, pp. 96–100.

[64] R. Estepa, J.E. Díaz-Verdejo, A. Estepa, G. Madinabeitia, How much training data is enough? A case study for HTTP anomaly-based intrusion detection, IEEE Access 8 (2020) 44410–44425.

[65] H.K.D.M. Bekkar, T.A. Alitouche, Evaluation measures for models assessment over imbalanced data sets, J. Inf. Eng. Appl. 3 (2013) 27–38.

[66] Snort, Network intrusion detection & prevention system, 2023, URL https://www.snort.org.

[67] A. Waleed, A.F. Jamali, A. Masood, Which open-source ids? Snort, suricata or zeek, Comput. Netw. 213 (2022) 109116.

[68] L. Rosa, P. Alves, T. Cruz, P. Simões, E. Monteiro, A comparative study of correlation engines for security event management, in: Iccws 2015-the Proceedings of the 10th International Conference on Cyber Warfare and Security, 2015, p. 277.

[69] M. Almgren, W. Aoudi, R. Gustafsson, R. Krahl, A. Lindhé, The nuts and bolts of deploying process-level ids in industrial control systems, in: Proceedings of the 4th Annual Industrial Control System Security Workshop, 2018, pp. 17–24.

[70] W. Ali, I.U. Din, A. Almogren, M. Guizani, M. Zuair, A lightweight privacy-aware iot-based metering scheme for smart industrial ecosystems, IEEE Trans. Ind. Inform. 17 (9) (2020) 6134–6143.

[71] W. Ali, I.U. Din, A. Almogren, B.-S. Kim, A novel privacy preserving scheme for smart grid-Based Home Area networks, Sensors 22 (6) (2022) 2269.

[72] Z. Ali, M.A. Shah, A. Almogren, I. Ud Din, C. Maple, H.A. Khattak, Named data networking for efficient iot-based disaster management in a smart campus, Sustainability 12 (8) (2020) 3088.

[73] R. Talat, M.S. Obaidat, M. Muzammal, A.H. Sodhro, Z. Luo, S. Pirbhulal, A decentralised approach to privacy preserving trajectory mining, Future Gener. Comput. Syst. 102 (2020) 382–392.

[74] A. Merlo, M. Migliardi, P. Fontanelli, Measuring and estimating power consumption in android to support energy-based intrusion detection, J. Comput. Secur. 23 (5) (2015) 611–637.

[75] G.A. Jacoby, N. Davis, Battery-based intrusion detection, in: IEEE Global Telecommunications Conference, 2004. GLOBECOM'04, Vol. 4, IEEE, 2004, pp. 2250–2255.